assigned to perform the tasks. Further, the customizable expression processing system is readily applicable to incorporate various forms of expression into various business and other applications.

The high-level, user friendly, easy to understand script language advantageously used to enter customizable expression decision logic shields users from the underlying complexities involved in incorporating a user defined expression within an application (such as involved in using JAVA or SQL, for example). It is also adaptable to meet different requirements and obviates any need to anticipate the many ways a user may formulate expressions to process data. The high level script language also facilitates the provision of a user interface that is straightforward to use by a member of the general public and yet provides the capability to implement any function that the user may desire.

The customizable expression processing system also supports an efficient method of maintaining and executing user entered customizable expressions in conjunction with associated business applications. This method advantageously involves version numbering expressions so that a latest expression version may be dynamically identified and compiled together with an associated business application. This is done to ensure execution speed of the composite application comprising the business application plus latest desired expression is not degraded. The version numbering system also supports the maintenance of multiple expressions as well as the use of an expression by multiple different applications and may also be used to ensure that a business application uses the same version of an expression throughout the lifetime of the business application. Further, the disclosed system enables restriction and control of the data that is exposed to a user for entry in an expression and supports association of descriptive information with items of data exposed to users. In addition multiple user selectable template expressions are provided for user alteration advantageously facilitating the user interface process.

The principles of the invention may be applied to any system and applications involving user entry of resolvable expressions to modify application function. Further, although the preferred embodiment is described as being implemented in JAVA, and XML this is exemplary only. The system may be implemented in a variety of different languages as the principles of the invention are independent of languages used.

In the system of Figure 1 a user enters an expression representing decision logic for use by a business application program 30 to tailor the application function to user requirements. An exemplary expression used to determine patient eligibility for reimbursement for a particular service is shown within item 579 of

Figure 6. The user enters the expression via a PC user interface 10 display menu associated with application interface 25 of application 30. The application interface 25 display menu provides windows supporting user selection of predetermined items 12 associated with application 30. Specifically, a user selects (from items 12) a predetermined template expression for amendment with predetermined data items and expression operators that are compatible with application 30 thereby advantageously simplifying the user interface. Application 30 employs one or more data specification XML files specifically associated with particular interface 25 of application 30. The data specification XML files are stored in an application 30 database and determine the predetermined data items 12 that are available for selection by a user via display windows of the PC user interface 10 display menu. The application 30 data specification files also optionally provide a user accessible description of individual exposed data items. Alternatively, in another embodiment a user may compose and apply an expression using PC interface 10 without using items 12.

Application 30 allocates both a Type and Group identifier to an expression entered by a user via the PC user interface 10 display menu associated with the application interface 25. The Type identifier identifies a specification used to define the input and output parameters that are received or provided by an application 30 procedure involved in processing the entered expression. An application such as application 30 typically comprises one or more procedures (termed objects) such as objects 31 and 33 of Figure 1. In more complex embodiments multiple application 30 procedures (e.g., objects 33 of Figure 1) may be involved in processing an entered expression. The expressions of a specific Type are compiled into a single JAVA class file. A JAVA class comprises a user defined data type and identifies the members of the class. The Group identifier allocated to an expression by application 30 identifies expressions of a particular Type that are compiled together into a single JAVA class file. Therefore expressions of a particular Type may comprise different Groups of expressions and a particular Type expression may be in more than one Group. Different Group identifiers may be allocated to a template (model) expression of a particular Type available for user selection and to a user entered expression of the same Type, for example.

A high level script language expression entered by a user via the PC interface 10 display menu associated with the application interface 25 is retained, together with a user entered identification name for the expression, in application server storage 14. Application 30 operating on the server initiates compilation of the high level script language expression into code of a particular JAVA class executable in conjunction with compiled application 30 in response to receiving the user entered

expression. In other embodiments, application 30 may initiate compilation using a different (i.e., non-Java) language compiler and upon other events including, for example, a convenient processing point when interruption of application 30 is not disruptive or upon collection of several expressions, for example. The entered expression is compiled into a JAVA class determined based on the Type and Group identifiers previously allocated to the entered expression by application 30. The resultant compiled expression is in the form of a JAVA class source file. Further, since application 30 is a JAVA application, application 30 and the compiled expression operate together with negligible degradation in speed. Application 30 initiates compilation of the entered expression using expression compiler 20 which in turn employs JAVA compiler 21 operating in conjunction with a SUN Microsystems tool JAVACC (a JAVA parser generator - unit 26) downloadable from Sun Microsystems web site with negligible degradation in speed associated with data marshalling.

Unit 20 applies the Type and Group identifiers previously allocated to the user entered expression by application 30 in compiling the expression into a particular selected JAVA class file. Further, unit 23 allocates a version number to the compiled JAVA class file so that a latest expression version may be dynamically identified and compiled together with business application 30 and associated application objects 31 and 33. This enables the latest compiled version of the expression to be identified and used at application 30 execution time. Thereby execution speed of the composite application comprising application 30 including objects 31 and 33 together with the compiled expression is not degraded. The resultant compiled JAVA class file is stored in accordance with allocated Type, Group and version attributes in unit 17 for access and loading of the latest expression version together with application object 31 (and other objects 33) upon initiation of execution of application 30.

In operation, application 30 calls an application object 31 procedure for evaluating a user created expression stored in unit 17 in response to user initiation of expression evaluation via interface 25. For this purpose application 30 employs the Type and Group identifiers associated, both with the application interface 25, and with the expression to be evaluated, for use in accessing this expression stored in unit 17. The identified compiled expression is therefore accessed from unit 17 and executed using data provided by application object 31. In this manner application 30 controls the linking of compiled JAVA class expressions in unit 17 with application objects 31 and 33 for creating a composite executable compiled application. Therefore application 30 governs the operation timing and use of created expressions.